REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 09/23/04	**2. REPORT DATE** Final Report	**3. DATES COVERED** *(From - To)* 05/01/01-04/30/04

4. TITLE AND SUBTITLE Building Interactive Digital Libraries of Formal Algorithmic Knowledge	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER N00014-01-1-0765
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Robert L. Constable	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Cornell University, Computer Science Dept. 4130 Upson Hall Ithaca, NY 14853	**8. PERFORMING ORGANIZATION REPORT NUMBER** 39544
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, Navy, Dept. of Defense 800 N. Quincy St. Arlington, VA 2217-5660	**10. SPONSOR/MONITOR'S ACRONYM(S)** ONR
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; distribution is unlimited

20041008 419

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This is a project to design and create a software system for sharing formal algorithmic mathematics among theorem provers, and for making formal algorithmic mathematics accessible to people who value verified accounts of algorithms. The project is also committed to creating interesting specimens of formally explained algorithms. Our work enables a new approach to CIP/SW; we call it information-intensive infrastructure protection. We describe the rationale for this approach in this report.

15. SUBJECT TERMS
Digital libraries, algorithmic knowledge, theorem proving.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPONE NUMBER** *(Include area code)*

Final Report

"Building Interactive Digital Libraries of Formal Algorithmic Knowledge"

Contract Number ONR N000014-01-1-0765

Reporting Period: May 1, 2001 – April 30, 2004

Submission Date: September 22, 2004

Prepared by:
Robert L. Constable
Cornell University
4130 Upson Hall
Ithaca, NY 14853

Summary

This is a project to design and create a software system for sharing formal algorithmic mathematics among theorem provers, and for making formal algorithmic mathematics accessible to people who value verified accounts of algorithms. The project is also committed to creating interesting specimens of formally explained algorithms. In the first year, we invested heavily in building a software infrastructure that includes a prototype Formal Digital Library (FDL) and procedures for storing formal content in it and procedures for presenting that content on the Web.

Our work enables a new approach to CIP/SW; we call it *information-intensive infrastructure* protection. We describe the rationale for this approach in this report.

This second year of the project has been a period during which the basic infrastructure and results established in the first year have borne visible fruit, and during which we have considerably enriched that infrastructure to support results anticipated in the third and critical year. During the next year (third) we expect to demonstrate our progress over two-and-a-half years and make a strong case for the optional two additional years of funding.

The most "visible" results are the collections of algorithmic knowledge posted on the Web from the Formal Digital Library (FDL). The collections are from three provers — MetaPRL, Nuprl, and PVS. For the Nuprl and PVS collections, we have harvested formal metadata whose value is directly apparent.

We have also demonstrated new direct access to the FDL using a new navigator tool and VNC (Virtual Network Computing). Extensive documentation and user manuals are available at the FDL Web page.

Among the collections are important new verified algorithms such as Red/Black trees, a small collection of graph algorithms, and a linear arithmetic package used in theorem proving. These provide the basis for illustrative Web-based articles that are semantically anchored in the FDL collections.

The additions to the infrastructure and basic capabilities over this period are not yet as visible. We have written a considerable amount of code for harvesting *formal metadata.* This code will be critical as we work this year to automate the Web-posting process. We have also explored mechanisms for formula-based search and for automatically clustering the FDL objects based on the latent semantics of the link structure.

We have extensively studied and explained fundamentals of "Logical Libraries" and how one might effect them. The two theoretical issues are, first, how to build a repository of certified (digitally expressed) knowledge as opposed to mere information, and second, how to enable different, sometimes incompatible, methods of certifying such knowledge to be accomodated in the same repository. Different clients may have radically different criteria for what counts as verified knowledge, and yet may be able to agree on substantial aspects and so share large parts of the knowledge repository. These issues are resolved by maintaining records of certification and strict accounting for bases of knowledge.

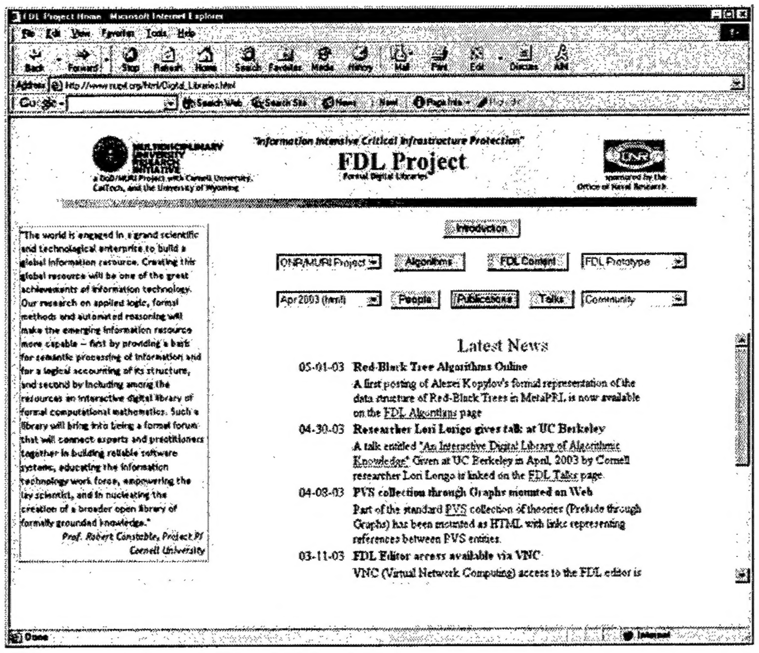
The "openness" of such repositories practically entails that criteria for certification include calls to agents external to the repository, and that methods for developing material for contribution to a logical library can be developed with the same accounting methods used in it, and that contributions not interfere with extant content (for example name collision must be avoidable).

Issues of cognitive accessibility include the attachment of informal explantory material to the logical material, provision of extra-logical organization of logical and other content, as well as utilities for exploiting the logical content in concert with the informal content. We continue to investigate

search based upon formula patterns, complementing search for words leading to content via concise annotations (see section below on Goals).
Larger social issues are discussed, including the cooperation of distinct repositories in ways that respect their independence and account for integrity of content composed from different sources, and the practicality of independent implementation of repositories to avoid an intolerable dependency on a few institutions.
We have continued our investigations of the theoretical foundation of inter–theory sharing, and the foundations for presenting a class of algorithms that is especially relevant to protecting the nation's critical software infrastructure, namely distributed algorithms and protocols. These foundational results were selected to complement investments being made by the Naval Research Laboratory in software engineering for reliability of distributed systems.
We have made significant progress towards practical methods of reflecting syntactic and computational aspects of logics. The bulk of the methods pertain to reasoning about expressions abstractly and are intended to be applied to the abstract syntax of typical contributions to the FDL itself. Logics sharing the FDL as a medium will then be able to refer to themselves and each other. This may be expected to expedite metamathematical work relating multiple logics, as it is hoped that it will provide a practical medium for enhancing various logics by reflection.
We expect to bring all of these threads together in the third year to demonstrate the power of information-intensive critical infrastructure protection. Our efforts are fundamental and progressive. That they are fundamental can be seen through their ties to two other MURI projects — SPYCE, and Language-Based Security — as well as through ties to research at the NRL. The fact that they are progressive can be seen from the fact that we are the only such project in the U.S., and we are highly competitive with European efforts that are far more extensive and well funded.

Project Web Page

We use the project Web page to post results, publications, lectures, briefings, algorithms and news items. We regard the Web page as an important supplement to this report. We will put the report on the Web page with hyperlinks. The page is at **http://www.nuprl.org/FDLproject/**.


FDL Project Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss AIM
Address http://www.nuprl.org/html/Digital_Libraries.html
Search Web Search Site News Page Info
Introduction
ONR/MURI Project
Algorithms
FDL Content
FDL Prototype
Apr 2003 (html)
People
Publications
Talks
Community
Done
Internet

MULTIDISCIPLINARY UNIVERSITY RESEARCH INITIATIVE
a DoD/MURI Project with Cornell University, CalTech, and the University of Wyoming

"Information Intensive Critical Infrastructure Protection"

FDL Project

Formal Digital Libraries

sponsored by the Office of Naval Research

"The world is engaged in a grand scientific and technological enterprise to build a global information resource. Creating this global resource will be one of the great achievements of information technology. Our research on applied logic, formal methods and automated reasoning will make the emerging information resource more capable – first by providing a basis for semantic processing of information and for a logical accounting of its structure, and second by including among the resources an interactive digital library of formal computational mathematics. Such a library will bring into being a formal forum that will connect experts and practitioners together in building reliable software systems, educating the information technology work force, empowering the lay scientist, and in nucleating the creation of a broader open library of formally grounded knowledge."

Prof. Robert Constable, Project PI
Cornell University

Latest News

05-01-03 **Red-Black Tree Algorithms Online**
A first posting of Alexei Kopylov's formal representation of the data structure of Red-Black Trees in MetaPRL is now available on the FDL Algorithms page

04-30-03 **Researcher Lori Lorigo gives talk at UC Berkeley**
A talk entitled "An Interactive Digital Library of Algorithmic Knowledge" Given at UC Berkeley in April, 2003 by Cornell researcher Lori Lorigo is linked on the FDL Talks page.

04-08-03 **PVS collection through Graphs mounted on Web**
Part of the standard PVS collection of theories (Prelude through Graphs) has been mounted as HTML with links representing references between PVS entities.

03-11-03 **FDL Editor access available via VNC**
VNC (Virtual Network Computing) access to the FDL editor is

Index of Accomplishments

- Basic FDL
  - Navigator manual
  - VNC interface
  - Additional collections
  - XML interface
    - PVS proofs
    - MetaPRL proofs

- Web-based Presentation of FDL Collections
  - PVS standard libraries
  - Graphs — PVS, Nuprl
  - Metadata harvesting
  - Formal metadata
  - Automating web presentation

- Creating New Content
  - Red/Black trees
  - Graph algorithms
  - Linear arithmetic
  - Distributed algorithms

- Foundational
  - Abstract object identifiers
  - Certificates and sentinels
  - Relating theories – thesis work

Reflection – thesis work (to be demonstrated at LICS '03)

Relationship to CIP/SW

Why is a formal digital library of algorithmic knowledge important to critical infrastructure protection? Here is one justification based on four categories of assertion: **A,** statements that are self evident; **B,** basic facts that have been discovered some time ago in computer science; **AB,** more modern discoveries in computer science; **C,** conclusions from these facts.
A: Algorithms, programs, processes and protocols are all examples of formal procedural knowledge. Because computers execute these procedures, systems of this kind of knowledge have become indispensable to modern society — to defense, health, learning, and discovery.
The nation's critical infrastructure includes a number of software systems, and elements of the physical infrastructure are controlled by software. Software is fundamentally systems of algorithms. To promote reuse of algorithms, to allow careful scrutiny of them and to preserve them, the research community has long supported the creation of libraries of code that can be shared, studied, improved, and preserved.
B1: Computer scientists and mathematicians since the Greeks have known that procedural knowledge is incomplete without corresponding factual knowledge. We know more about an algorithm than that it ran with certain results. Algorithms are produced in concert with factual and analytical knowledge that determines their design and justifies claims of the designers and programmers that the algorithms accomplish the tasks for which they are intended. This knowledge is closely related to computational and constructive mathematics. Not all of this critical knowledge is written or saved in any form, and even the elements which are carefully written can easily become lost or disconnected from the algorithms as they evolve. Consequently, further development of such algorithms is more likely to be erroneous.
Collecting declarative knowledge along with the algorithms and linking it to the code is considered to be excellent professional practice by both the research community and industry. When code is collected into libraries, associated declarative knowledge should be included as well so that it can be scrutinized, criticized, improved, checked, and so that it will evolve with the code and be preserved with it.
This practice is known to be effective. Moreover this declarative knowledge is frequently critical to understanding the algorithm, as we can tell from the way textbooks present algorithms. However, code libraries do not typically include the amount of detail given by textbooks, when in fact we believe that even more such knowledge is necessary to justify the code and explain it.
AB2: Significant parts of the declarative knowledge documenting algorithms can now be formalized. Because computers can check and process this formal declarative knowledge, it has become included as a part of the code verification process.
As computer security and software reliability become more important to government and industry, formal code documentation and its verification will increase. Libraries containing formal documentation and explanation will then become indispensable to society – to defense, health, learning and discovery.
The classification of knowledge into computer checked (formal) and noncomputer checked (informal) aids those who are responsible for locating flaws in reasoning that justifies algorithms in critical software systems. It is vastly less likely that errors occur in the computer checked knowledge.

For subtle algorithms or tedious ones with many cases, experience has shown that computer assistance in the form of extended type checking, proof checking and model checking is essential to finding errors in reasoning. In the case of concurrent and distributed algorithms, even simple protocols can be so subtle and complex that it is very difficult to program them correctly without computer assistance in checking for errors or automatically creating arguments for correctness along with the code.

AB3: Libraries of declarative knowledge require trustworthy mechanisms that account for long chains of logically connected evidence. Surprisingly little is known about these accounting mechanisms for any feasible means of providing sufficiently large and scalable collections of formal declarative knowledge.

Even less is known about mechanisms that can account for the formal evidence produced by different verifiers with incompatible logics. Nevertheless, there is good reason to believe that there will always be several verifiers in use because there are several incomparable logics with which to carry out verification tasks. Just as there are many different programming languages and dialects, each one suited to a certain class of problems, there are also many different logical systems for verifying declarative knowledge, each well suited to a certain class of problems. Furthermore, research develops new approaches that are not simple modifications to old ones, and can be expected to continue that way. For example, linear logic might become a practical formalism, various mixtures of types, sets, and domains might fit into place in unforeseen ways, and we might use relevance logic to contain inconsistencies so that they do not spread beyond local effects.

Formal logics are extremely precise; minor changes in a single rule can render the entire logic inconsistent. There are many points at which all the modern logics for theorem proving differ. Some allow empty types, some do not. Some depend on decidable type systems, others do not. Some allow dependent types, others do not. Some use the axiom of choice some do not, and among those that do, some use the Hilbert epsilon operator to state it, and others do not. Some logics allow full recursive types, others only restricted recursion. In some logics types are ordinary objects, in other logics they have a special limited status.

C1: This ONR/MURI project is providing the required technical understanding needed to build large libraries of formal declarative knowledge in digital form and account for evidence archived in them. We are applying this understanding to building small sustainable examples, especially examples that support multiple distinct verifiers.

C2: In addition to mechanisms to account for long chains of justifications, these libraries require standard services — organizing, archiving, and searching. The formal character of this knowledge opens new possibilities for computer assistance in these tasks. We are exploring those possibilities and will test them on our examples.

B4: Using computers to check and even generate declarative knowledge remains a difficult task. The rate at which formal knowledge can be generated depends on the power of the logical reasoning tools, called provers, and on the amount of formal knowledge available to the provers. It also depends on the number of trained personnel. This crude equation illustrates the relationship.

$$\mathbf{verification\_rate} = \mathbf{prover\_strength} \times \mathbf{size(knowledge\ base).}$$

Researchers have spent 30 years building powerful provers and sharing algorithms used in them, and they have spent no time making large collections of formal knowledge that can be shared. Government and industrial funding has helped create the powerful provers but there has been essentially no funding for the the knowledge base. ONR/OSD is a leader in this regard.

As formal knowledge is shared and made available to provers, the verification task becomes faster and easier. Collecting this knowledge into libraries facilitates sharing, criticism, and improvement. The libraries will also contribute to the education of professionals able to use verification tools and produce formal content.

C3: Our project creates mechanisms that allow logically sound sharing of formal knowledge. We are providing a basis for creating a vast collection of formal facts. It is plausible that once a critical mass of formal knowledge is assembled in a form allowing automatic sharing, a singularity will occur in the capacity of the research community to produce more.

C4: If this singularity results in the rapid creation of formal knowledge about algorithms, the benefits to society will be enormous because that capability will enable a dramatic increase in the reliability and security of software, and it will free a large number of highly trained people to focus their efforts more productively.

It is possible that a technology for the routine verification of formal knowledge will also dramatically alter the means of verifying and communicating precise knowledge of many kinds.

The stability, accessibility and extensibility of libraries of formal knowledge, is key to harnessing the power of a community of developers who use results of the formal knowledge providers. Libraries of formalized knowledge form a basis for long-term collaboration between parties with differing interests and skills in the development of such knowledge.

Goals

1. Foundations of Logical Libraries

   Stuart Allen's notes on the structure of the Formal Digital Library, Notes on the Design and Purpose of the FDL, lay out a foundation for the notion of a logical library described in the project Goals section of the Web page. Work on this topic progresses as we approach implementing the capabilities described in Allen's notes.

2. Formal System Cooperation

   2.1. Create model-sharing environment for the Logic of Events (LoE) (Nuprl, MetaPRL, PVS, JProver combined)

   For critical infrastructure protection, distributed algorithms are very important. These algorithms are also investigated by the Software Engineering section at the Naval Research Laboratory using the PVS theorem prover in the Timed Automata Modeling Environment (TAME). We want to demonstrate the value of the FDL in algorithm development by combining the capabilities of PVS, MetaPRL and Nuprl through the Formal Digital Library.

   2.2. Installing beta-version of sentinels in FDL

   It is not possible to simply combine results from different theorem provers. The exact conditions under which a combination is possible is a fundamental matter for logical libraries such as the FDL. One approach to keeping track of logical dependencies is presented in our foundational work on the FDL. We plan to concretely illustrate this mechanism on various shared libraries such as number theory, lists, graphs, trees, and protocols.

   2.3. Metamathematics FDL theory – basis for relating theories

The basic logical results allowing sharing among Nuprl, HOL, and PVS have not been formally supported. We intend to lay the ground work for this by linking theoretical results to the sentinels mentioned above.

2.4. Formal Symbolic Algebra

Computer algebra systems are a substantial source of basic algorithmic mathematics. We are making a small effort to connect some of these algorithms to the provers. This is a major activity in Europe. Our approach is to use the module system of the MetaPRL logical framework to track the domains of a system such as Axiom. The module system and Kopylov's dependent records are both important for organizing the theory. The module system manages system content, like rules and tactics, and the records manage the formal parts like groups, rings, and fields.

2.5. Preparing to support Larch proofs

The corporation ATC-NY has allocated funds to transfer a large number of Larch theorems to the FDL. They are also interested in restoring their Larch prover within the FDL. They have done some preliminary work in this direction, and they plan more when we are ready.

Larch is a language with overloading so after parsing there is a "sort-checking" phase that assigns a sort to every term and chooses a signature for every function symbol. Before we can do any semantic analysis (in particular the checking of proofs) of the Larch theories we have to get them sort-checked and store in the library a version with all function symbols resolved. So the first tool we must build is the sort checker. This is non trivial because Larch has a complex syntax for specifying how traits include other traits with renaming, and that defines the possible signatures for the function symbols.

2.6. Verified decision procedure for abstract algebra and arithmetic

The MetaPRL system is building a very general arithmetic decision procedure. The procedure is being used to derive results in abstract algebra. As the algebra work progresses, it will be used to derive more general axioms for the decision procedure. Elements of this work will be included in the FDL and can be contributed to QPQ as well. Since the procedure generates primitive proofs, it does not need to be verified directly.

3. Collections

3.1. Automating PVS acquisition and posting

Currently the methods used for acquisition of PVS libraries have involved an amount of human supervision that is not practical for repeated efforts of acquiring PVS files for the FDL. The methods should be completed so that no human intervention is required beyond targeting the PVS files.

The acquisition of PVS proofs requires significantly more human intervention because the PVS proof engine often crashes during the process. We must either find a more reliable method of running PVS for proof acquisition, or develop a fault-tolerant method.

Currently although a large number of proofs have been collected into the FDL, they have not been included in our web presentations. Methods for presenting them have yet to be finished, and more incremental methods for managing them as data are required. We must also account for proofs we have been unable to acquire.

A final stage of this work is to verify that the automatic acquisition is correct by replaying the proofs in the FDL.

The entire acquisition, posting and verification work is very compute intensive.

3.2. Collecting formal metadata for PVS libraries

While establishing links between objects is fundamental to acquisition for the FDL, it is quite important to assemble data from those objects to forestall some of the web walking clients would otherwise have to do themselves. For example, it is of general interest among clients which objects refer directly or perhaps indirectly to a given object. In addition to adding such cross-referencing indices to the FDL itself, there are further issues of organization to make them surveyable by human browsers of the Web presentation, since the indices can become long.

Extending the indices for cross-referencing to provide elaborate larger organizations of objects is a further design and programming problem. This can involve the collection of related objects such as all lemmas needed for a proof.

It has been particularly challenging to collect metadata for PVS because there is no notion of primitive proof. We expect to encounter this problem with other proof systems as well.

3.3. Preparing to support Larch proofs

Larch is a language with overloading so after parsing there is a "sort-checking" phase that assigns a sort to every term and chooses a signature for every function symbol. Before we can do any semantic analysis (in particular the checking of proofs) of the Larch theories we have to get them sort-checked and store in the library a version with all function symbols resolved. So the first tool we must build is the sort checker. This is non trivial because Larch has a complex syntax for specifying how traits include other traits with renaming, and that defines the possible signatures for the function symbols.

3.4. Other libraries

We might be able to add HOL and Minlog libraries pending technical discussions over the summer, and we are alert for other opportunities, e.g. a possibility with Mizar.

4. Access to Content

Access Media

4.1. Posting to other presentation media (Helm, OMDoc, MathML)

We have connected Nuprl libraries to Helm and are discussing connections to OMDoc and MathML. We plan to make it possible to connect all FDL content to these presentation resources.

4.2. Dynamic FDL server architecture

Full access to the formal knowledge content is available only at the FDL server and only using the editors and navigators provided for the FDL. Here there is complete access to proofs as well as theorems, definitions, and algorithms. Execution of algorithms is principally in ML, Lisp, and OCaml.

There are currently two ways to access the FDL via web servers: (1) a limited demonstration utility showing that XML formatted objects can be retrieved from the FDL, and (2) projections of material from the FDL intended to show to readers the kind of content available. Three advances to be made are giving a full access to the FDL via a server; supplementing the precomputed projections for browsing with a companion server for dynamic browsing requests too expensive to precompute generally; dynamically tying the relation between the browser presented information and the active FDL more directly, i.e., making the browseable material more a "view" than a "projection."

Semantic Access

4.3. Semantic anchoring of expository text
We will produce examples of expository text that are "semantically anchored" in the formal proofs and definitions of the FDL that support it. The first example of a semantically anchored research paper will be one about Event Systems, to which great value can be added by linking it to the extensive FDL material corresponding to the subject matter of that paper.

4.4. Concise Annotation of FDL content
Work is underway for developing methods to add concise annotations to FDL objects. The purpose of the annotations is to provide brief paraphrases in words for formal concepts and entities of the FDL. These complement more expository texts for readers and also serve as a more focussed basis for word-based search since each concise annotation is about fewer subjects. Further, these concise annotations are considerably less expensive to produce by knowledgeable annotators than are broader expository texts; they are simpler technically to produce since they consist simply of ordinary text, and are therefore amenable to elementary creation through forms on the Web.
Although it is of little intellectual interest, we must provide some recommendations to readers as to which word searches are likely to bear fruit. This could be as simple as an index of prepared search buttons.

4.5. Continued exploration of formula-based search
Concise annotations (as well as semantically anchored texts) serve as entry points via word-search to the FDL contents. But once one has the formula structure in hand, one has a radically different means for specifying search criteria. For example one could specify what it means to be a theorem expressing distributivity of any binary operation over any other binary operation. Development of methods for specifying useful patterns is underway, with particular focus on automatically identifying operators that can be considered interchangeable for most purposes based upon mining theorems relating them.

5. Formal Content Creation

5.1. Create model-sharing environment for the Logic of Events (LoE) (Nuprl, MetaPRL, PVS, JProver combined)
For critical infrastructure protection, distributed algorithms are very important. These algorithms are also investigated by the Software Engineering section at the Naval Research Laboratory using the PVS theorem prover in the Timed Automata Modeling Environment (TAME). We want to demonstrate the value of the FDL in algorithm development by combining the capabilities of PVS, MetaPRL and Nuprl through the Formal Digital Library.

5.2. General content production

A significant part of the overall project effort is the production and integration of new content. Some of this arises as preexisting libraries are moved into the FDL, annotated, and posted. As part of this process, formal metadata is collected and displayed.

Other content arises as we expand the scope or depth of the existing collections, e.g. enhancing graph theory, adding new algorithms such as red/black trees, and adding new theories such as the logic of events. In all cases the libraries are integrated into the FDL, and they are used to test the acquisition and posting capabilities.

5.3. Verified decision procedure for abstract alg and arith

The MetaPRL system is building a very general arithmetic decision procedure. The procedure is being used to derive results in abstract algebra. As the algebra work progresses, it will be used to derive more general axioms for the decision procedure. Elements of this work will be included in the FDL and can be contributed to QPQ as well (see Community on the FDL home page). Since the procedure generates primitive proofs, it does not need to be verified directly.

6. Publication and Communications

6.1. Papers, lectures and conference preparation

We are writing a variety of papers about the FDL, and R. Constable has again been invited to speak about this work in Europe. Much of June and July will be occupied in preparing articles and presentations, which will appear on the Web site in due course.

The Web site keeps a running account of the publications.

6.2. FDL technical meetings (OMDoc, Peer-to-peer, PVS automation)

We have plans to work with Michael Kohlhase on OMDoc and MathML over the summer. We will also work with C. Jechlitschek in Ithaca this summer. We hope to have a meeting with a PVS staff member to help us automate the PVS acquisition process.

7. Graduate Student Supervision

The professors and research assistants are heavily involved in supervising the work of the graduate students. In some cases involving implementation, this requires several hours per week.

8. System Support

Maintaining and improving the FDL involves a great deal of basic system support. It is necessary to operate several theorem provers, (JProver, MetaPRL, Nuprl, and PVS for now; others later) and relate them. We also support a VNC connection to the FDL.

Publications

Dependent Intersection: A New Way of Defining Records in Type Theory, by A. Kopylov. LICS, 2003.

The FDL Navigator: Browsing and Manipulating Formal Content, by C. Kreitz. Technical Report, Cornell University, 2003.

Introduction to the Objective Caml Programming Language, by J. Hickey. California Institute of Technology, 2003.

A Logic of Events, by M. Bickford and R. Constable. Technical Report, Cornell University, 2003.

MetaPRL - A Modular Logical Environment, by J. Hickey, et al. Technical Report, California Institute of Technology and Cornell, 2003.

Practical Reflection in Nuprl, by E. Barzilay, S. Allen, and R. Constable. Presentation at LICS '03.

Abstract Identifiers, Intertextual Reference and a Computational Basis for Recordkeeping, by Allen, S. First Monday, vol. 9, no. 2, 2004.

Formal Design Environments, by B. Aydemir, A. Granicz, and J. Hickey. TPHOLs, 2002. Appears in NASA Technical Report NASA/CP-2002-211736, 2002.

Notes on the Design and Purpose of the FDL, by S. Allen. Cornell University, ongoing.

Reflecting Higher-Order Abstract Syntax in Nuprl, by E. Barzilay and S. Allen. TPHOLs, 2002.

Representing Nuprl Proof Objects in ACL2: toward a proof checker for Nuprl, by J. Caldwell and J. Cowles. ACL2 Workshop, 2002.

Theory and Implementation of an Efficient Tactic-Based Logical Framework, by Aleksey Nogin. PhD Thesis, Cornell University, 2002.

Steps Toward a World Wide Digital Library of Formal Algorithmic Knowledge, by R. Constable, S. Allen, M. Bickford, J. Caldwell, J. Hickey, and C. Kreitz. Unpublished manuscript 2003.